colloquium exactarum

# ANÁLISE DE TÉCNICAS DE DETECÇÃO DE COLISÃO BROAD PHASE EM APLICAÇÕES GRÁFICAS INTERATIVAS BIDIMENSIONAIS EM UM AMBIENTE WEB

## ANALYSIS OF TECHNIQUES OF BROAD PHASE COLLISION DETECTION IN INTERACTIVE TWO-DIMENSIONAL GRAPHICS APPLICATIONS IN A WEB ENVIRONMENT

Victor Hugo de Paiva Gonçales, Francisco Assis da Silva, Mário Augusto Pazoti, Danillo Roberto Pereira, Leandro Luiz de Almeida

Universidade do Oeste Paulista – UNOESTE, Faculdade de Informática – FIPP, Presidente Prudente, SP
e-mail: victorvhpg@gmail.com, {chico, mario, danilopereira, llalmeida}@unoeste.br

**RESUMO -** Ambientes gráficos interativos requerem um alto nível de precisão para poder simular comportamentos físicos em tempo real, um deste comportamento é a colisão entre objetos. Realizar a detecção de colisão em tempo real exige um alto custo computacional. Uma técnica de colisão mal aplicada pode comprometer todo o desempenho da aplicação. Este trabalho apresenta técnicas utilizadas na detecção de colisão de objetos 2D num ambiente gráfico interativo e faz uma análise experimental do desempenho dessas técnicas na fase *broad phase* com a técnica de Grid e Força Bruta em um ambiente Web, com o propósito de determinar em quais situações pode-se ter um melhor ou pior desempenho. Com os experimentos realizados, foi possível identificar que a técnica de Grid possui um melhor comportamento em relação à técnica de Força Bruta, obtendo um desempenho satisfatório num ambiente Web, principalmente quando a quantidade de objetos no ambiente é muito alta.
**Palavras-chave:** computação gráfica; detecção de colisão; *broad phase;* Web.

**ABSTRACT -** Interactive graphics environments require a high level of precision in order to simulate physical behavior in real time, one this behavior is the collision between objects. To accomplish real time collision detection requires a high performance cost. A misplaced collision technique can compromise the entire application performance. This paper presents techniques used in collision detection of 2D objects in an interactive graphical environment, and makes an experimental analysis of the performance of these techniques in broad phase with the Grid technique and Brute Force in a Web environment, in order to determine in what situations can themselves have a better or worse performance. With the experiments carried out it was possible to identify that the Grid has a better performance against Brute Force technique, obtaining a satisfactory performance in a Web environment especially when the number of objects in the environment is too high.
**Keywords:** computer graphics; collision detection; broad phase; Web.

Recebido em: 30/07/2015
Revisado em: 13/06/2016
Aprovado em: 01/09/2016

## 1. INTRODUÇÃO

Aplicações gráficas interativas simulam comportamentos físicos do mundo real, e um desses comportamentos é a colisão. De acordo com Ericson (2005), a detecção de colisão determina se dois objetos estão em contato, onde ocorre o contato e quando ocorre. Para isso, são utilizados algoritmos de detecção de colisão, mas esse processo pode não ser tão simples e tem sido um desafio das aplicações gráficas interativas (SANTOS, 2007).

A detecção de colisão é utilizada em vários tipos de aplicações gráficas, incluindo jogos eletrônicos, ambientes de simulação, robótica e protótipos virtuais. Algumas dessas aplicações necessitam de uma detecção de colisão em tempo real, que seja muito eficiente, por exemplo, jogos eletrônicos envolvem simulações que requerem uma grande quantidade de verificações de colisões, que são efetuadas em torno de 30 a 60 frames por segundo (FPS). Um sistema de colisão mal projetado em um jogo pode ser um grande gargalo no desempenho da execução do jogo (ERICSON, 2005).

Em ambientes virtuais as colisões devem ser precisas para que a interação do usuário seja tão interativa e real quanto ao mundo em que vivemos, garantindo uma imersão real no ambiente interativo (TADDEO, 2005).

Tendo em vista que as técnicas de colisão podem ser um grande gargalo de desempenho nas aplicações gráficas interativas, este trabalho faz uma análise experimental de técnicas de detecção de colisão na fase *broad phase*, onde foram analisadas as técnicas de Força Bruta e Grid utilizando *Spatial Hashing*. Para realizar a análise, foi implementado um ambiente gráfico 2D interativo sendo executado em um navegador Web.

Esse trabalho contribui com uma análise das técnicas de detecção de colisão na fase *broad phase* para ambientes bidimensionais em um ambiente Web, identificando em quais situações determinada técnica de detecção de colisão pode ter um melhor desempenho que outra e identificar qual a melhor estratégia para se obter um melhor aproveitamento num ambiente Web. Buscou-se analisar a influência da quantidade de objetos em relação ao desempenho, identificar a melhor estratégia de divisão do espaço em grid, obter um melhor desempenho na detecção de colisão que não comprometa uma aplicação interativa em tempo real num ambiente Web, e analisar a influência do envoltório tipo esfera e tipo AABB na fase *broad phase*.

As demais seções deste trabalho estão organizadas da seguinte maneira: na Seção 2 são apresentados os conceitos e técnicas de detecção de colisão na fase *broad phase*; na Seção 3 é apresentada a metodologia utilizada no desenvolvimento do protótipo para a execução dos experimentos; na Seção 4 são

apresentados os experimentos realizados; e por fim na Seção 5 são apresentadas as considerações finais deste trabalho.

## 2. CONCEITOS FUNDAMENTAIS

Um método normalmente utilizado no processo de detecção de colisão é efetuar a detecção em duas fases distintas, a primeira chamada *broad phase* e a segunda chamada de *narrow phase* (SOUZA, 2014).

Na fase *broad phase* são identificados os possíveis grupos de objetos que poderão colidir e os que definitivamente não vão colidir. O objetivo é agrupar os objetos que estão próximos, já na fase *narrow phase* é realizado o teste de colisão detalhado entre os objetos que foram identificados na fase anterior (ERICSON, 2005).

Na fase *broad phase* não é necessário conhecer a forma geométrica real do objeto, pois ele é envolvido numa forma mais simples chamada envoltório, que normalmente pode ser uma esfera ou um retângulo conhecido como *axis-aligned bounding box* (AABB) (SOUZA, 2014).

De acordo com Schornbaum (2009), o uso de envoltórios faz com que os testes de potenciais colisões sejam mais rápidos, devido a simplicidade de calcular a intersecção entre eles, não sendo necessários cálculos mais custosos como no caso de intersecção de formas geométricas mais complexas.

A Figura 1(a) mostra um conjunto de objetos com formas variadas, sendo mais complexos em relação a objetos simples como esfera ou retângulo, o que torna os cálculos mais custosos para determinar a intersecção entre eles. A fim de agilizar os cálculos de intersecção, esses objetos são envolvidos em formas mais simples, como uma esfera (Figura 1(b)) ou um retângulo AABB (Figura 1(c)).


**Figura 1**. Objetos envolvidos com formas mais simples.
Fonte: (SCHORNBAUM, 2009)

Envoltórios esféricos são representados por um ponto central e um raio, eles se intersectam quando a distância entre seus centros é menor ou igual a soma de

seus raios, já os AABB podem ser representados por dois vértices contendo a maior e a menor coordenada respectivamente, eles se intersectam quando todos seus vértices possuem uma interseção entre eles (SCHORNBAUM, 2009).

A detecção de colisão entre envoltório de objetos na fase *broad phase* pode ser feita de várias maneiras, sendo que a comparação de todos os objetos com todos os pares possíveis é conhecida como Força Bruta, considerada uma técnica bastante limitada devido ao alto número de comparações (ROCHA E RODRIGUES, 2007).

A fim de diminuir a quantidade de testes de colisões, uma das técnicas utilizadas na fase *broad phase* é o particionamento do espaço, onde o ambiente em que os objetos estão é dividido para se criar uma estrutura em que os objetos são agrupados de acordo com suas posições (KERA*;* PEDRINI; NUNES, 2007).

O particionamento do espaço pode ser feito usando uma grid com células uniformes ou pode ser dividindo o espaço numa estrutura hierárquica, onde o espaço é dividido recursivamente (ERICSON, 2005).

Grid é uma maneira de dividir o espaço do ambiente onde estão os objetos. Nessa técnica o espaço é dividido em células uniformes, então cada objeto é associado à célula em que está contido. Isso faz com que o teste de colisão só seja feito entre os objetos que compartilham a mesma célula, sendo que um mesmo objeto pode ocupar mais de uma célula (ERICSON, 2005).

Na Figura 2 pode ser visualizado como fica o espaço particionado em grid com 4 células uniformes.

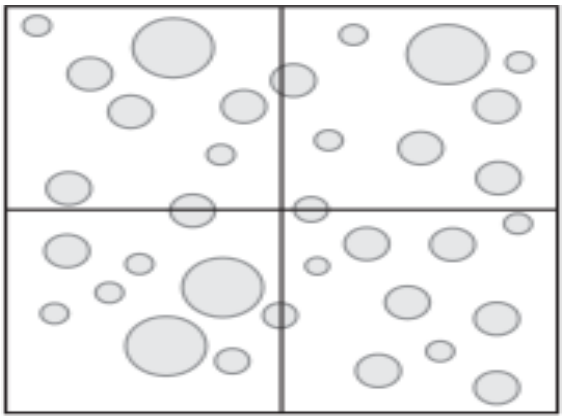

**Figura 2**. Particionamento do espaço em grid.
Fonte: (ERICSON, 2005).

Na Figura 3 é apresentada uma grid com os objetos envolvidos em um retângulo *axis-aligned bounding box* (AABB). Observa-se que um objeto pode ocupar mais de uma célula simultaneamente.

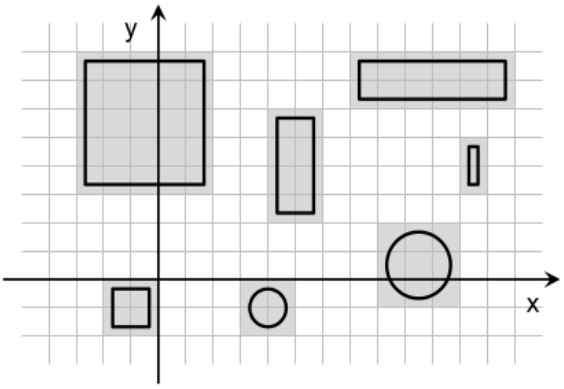


**Figura 3**. Grid com objetos ocupando mais de uma célula.
Fonte: (SCHORNBAUM, 2009).

De acordo com Souza (2014), comumente o tamanho das células de uma grid deve ser o tamanho do maior objeto presente no espaço, fazendo com que um

objeto possa ocupar no máximo quatro células num ambiente bidimensional.

Uma técnica de grid que foi utilizada por Hastings, Mesit e Guha (2005) é a *Spatial Hashing*, que de acordo com os autores, nessa técnica os objetos são projetados em um vetor de registro de uma dimensão conhecido como tabela *hash,* onde cada índice é gerado de acordo com a disposição dos objetos no ambiente, permitindo uma rápida localização e detecção. Para implementar *Spatial Hashing*, pelo menos 3 itens são necessários: uma grid, uma funcão *hash* e uma tabela *hash*.

A grid deve ser gerada de acordo com o tamanho do espaço e tamanho da célula, a função *hash* retorna o índice da célula da grid de acordo com uma posição 2D informada e a tabela *hash* é onde ficam os objetos armazenados, agrupados de acordo com a célula em que estão contido, sendo que cada índice da tabela *hash* é uma identificação da célula (HASTINGS; MESIT; GUHA, 2005).

Na Figura 4 é possível observar como os objetos são agrupados na tabela *hash*.

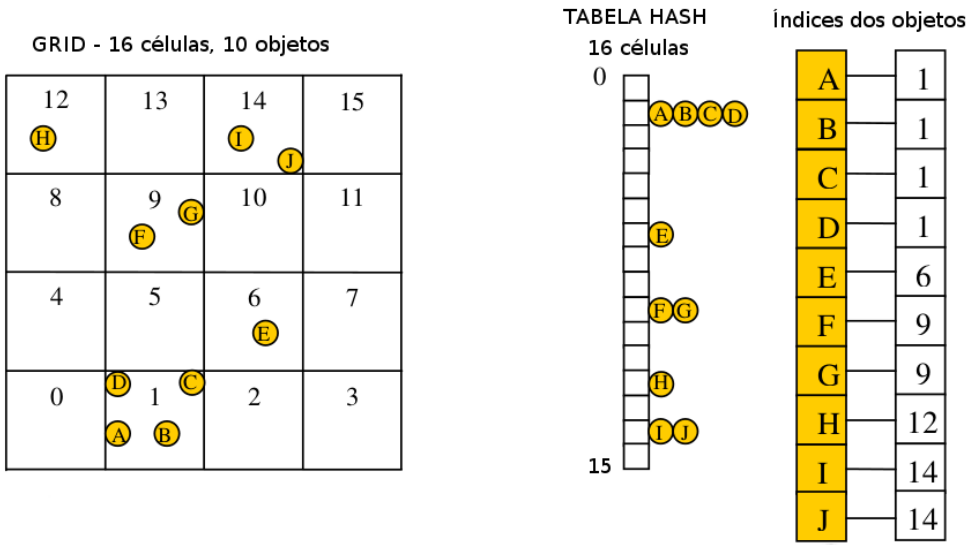


**Figura 4**. Particionamento do espaço em grid com *Spatial Hashing*.
Fonte: (HASTINGS; MESIT; GUHA 2005).

Hastings, Mesit e Guha 2005 aponta que tendo as células maiores que o tamanho médio dos objetos, obtém-se um maior desempenho em *Spatial Hashing*, contudo, as células devem ser menores quando comparadas ao tamanho do espaço onde estão todos os objetos. Os autores também observaram que células menores (até certo ponto) resultam num desempenho maior, mas causam mais uso de memória, pois cada objeto acaba ocupando mais de uma célula.

## 3. METODOLOGIA

Nessa seção são apresentadas as tecnologias e a metodologia que foram empregadas para o experimento realizado. As

técnicas para a detecção de colisão, assim como todo o ambiente, foram implementadas na linguagem JavaScript. Foi implementado um ambiente gráfico 2D interativo onde foram criados vários objetos de tamanho e velocidade de maneira aleatória, respeitando valores mínimos e máximos, estes objetos se movimentam de forma aleatória. Após a geração dos objetos, foi aplicada a análise das técnicas.

A cada frame, uma colisão é detectada quando um objeto intersecta o outro.

Tendo em vista que este trabalho analisou apenas a fase *broad phase*, os testes de colisão na fase *narrow phase* foram feitos entre seus envoltórios já que os objetos foram criados na forma de seu próprio envoltório.

Na técnica de Força Bruta os testes de colisão foram feitos entre todos os objetos contra todos.

Na técnica de Grid foi utilizado o tipo *Spatial Hashing,* conforme Hastings, Mesit e Guha (2005). Foi informada a largura, altura do ambiente e o tamanho da célula a partir destas entradas foi utilizada a técnica de *Spatial Hashing,* sendo que a quantidade de células na horizontal foi obtida dividindo a largura do ambiente pelo tamanho de uma célula, já a quantidade de células verticais foi obtida dividindo a altura do ambiente pelo tamanho da célula. Com isso foi criado uma tabela *hash* onde cada índice desta tabela representava uma célula.

Para determinar quais índices um objeto ocupa na tabela *hash*, foi utilizada uma função *hash* que a partir de um ponto 2D retorna o índice na tabela *hash*. Levando em consideração que um objeto possui mais de um ponto, então para obter os índices foram utilizados pontos chave do objeto, sendo que para objetos com envoltório AABB foram utilizados os 4 pontos extremos do objeto, como mostra Figura 5.

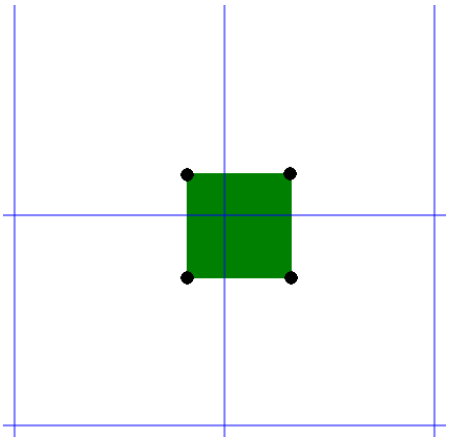

**Figura 5.** Pontos chave em um objeto com envoltório AABB.

Em objetos com envoltório tipo esfera os pontos chaves são os pontos extremos a partir do centro do objeto (Figura 6) conforme Hastings, Mesit e Guha (2004).

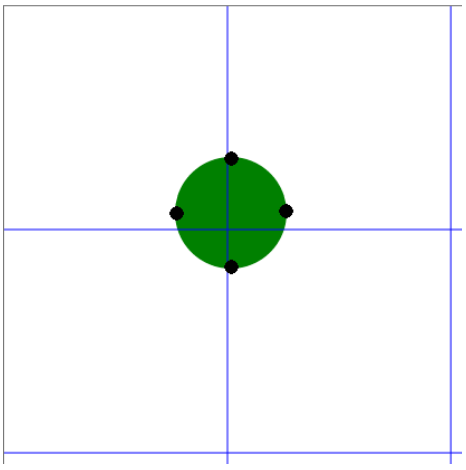

**Figura 6**. Pontos chaves em um objeto com envoltório AABB.

Cada posição da tabela *hash* é uma lista, podendo ter mais de um objeto indexado.

Nos testes de colisão, foi percorrido cada índice da tabela *hash* e realizado a detecção de colisão entre os objetos que estavam no mesmo índice.

O experimento foi executado num ambiente Web, em que foi utilizado um navegador Web para a execução do experimento.

A análise dos resultados foi baseada nos valores obtidos em cada técnica. Para cada análise foram parametrizadas algumas entradas, como tamanho do ambiente, total de objetos que serão gerados, valores mínimos e máximos para o tamanho e velocidade dos objetos, tipo de envoltório geométrico (esfera ou AABB), sendo que na técnica de Grid foi possível definir o tamanho das células da grid.

Foram analisados os seguintes itens:

- Total de testes de colisão: para cada frame será quantificado o número de testes de colisão entre os objetos;
- Frames por segundo (FPS): quantidade de vezes que o ambiente é desenhado por segundo;
- Total de objetos no ambiente;
- Tempo gasto na detecção de colisão: para cada frame é mensurado o tempo gasto nos testes colisão.

Através destas análises foi possível determinar qual técnica possui um melhor desempenho.

## 4. EXPERIMENTOS

Para a realização dos experimentos foi construído um ambiente gráfico 2D interativo implementado na linguagem JavaScript, onde foi possível parametrizar alguns dados de entrada para geração do ambiente. Na Figura 7 é apresentada a tela de parametrização, onde são definidos todos os parâmetros para a construção do ambiente 2D.

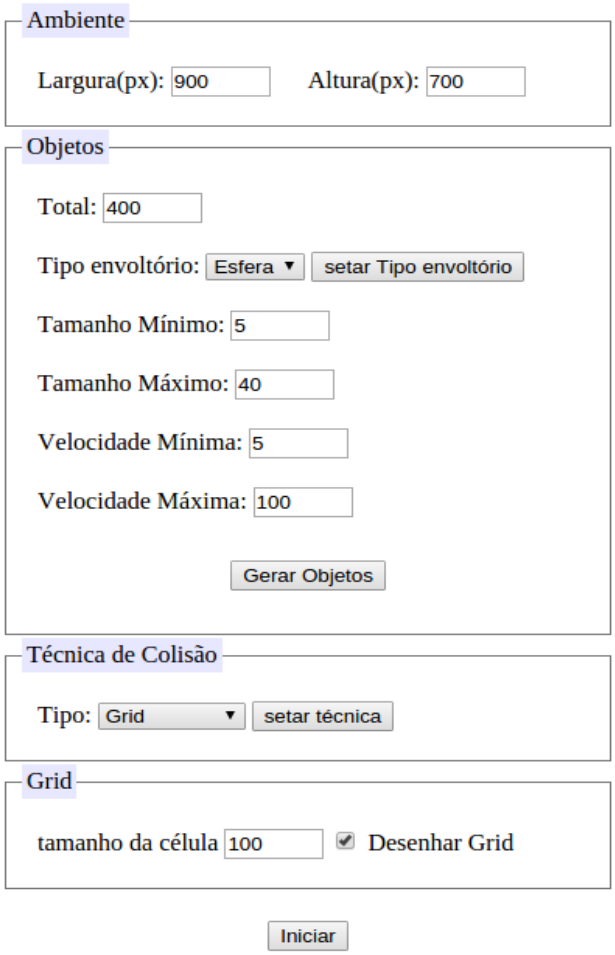


**Figura 7**. Tela de entrada de parâmetros do ambiente.

Na Figura 8 é apresentado o ambiente gerado, que é composto por objetos que se movimentam em direções aleatórias, verificando se há colisão com algum outro objeto. Quando é detectada uma colisão os objetos colidentes ficam na cor vermelha.

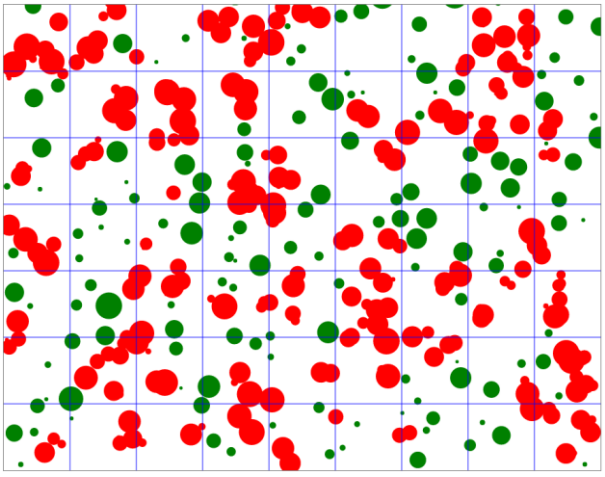

**Figura 8**. Ambiente 2D, com objetos em movimento.

Levando em consideração que a cada frame o desempenho não é constante, cada valor obtido na análise foi obtido calculando a média desse valor após a execução de 500 frames. O tamanho do ambiente foi fixado em 2000x2000 pixels.

Os experimentos foram executados em um navegador Web (Chrome 43), sendo executado em um notebook com a seguinte configuração:

- Sistema Operacional: Ubuntu 15.04 (64 bits);
- Processador: Intel® Core™ i7-4500U CPU @ 1.80GHz × 4;
- Memória RAM: 8GB.
- Placa de Vídeo AMD Radeon® HD 8670M, 2GB de memória DDR3;

## 4.1. ANÁLISE 1

Na primeira análise, foi verificada a influência do tamanho da célula da Grid em relação à taxa de frames e tempo gasto na fase de colisão entre todos os objetos. Para esta análise foram gerados 1000 objetos no ambiente, cada objeto com tamanho fixo de 50x50 pixels, direção e velocidade aleatória (de 5 a 50 pixels por segundo), utilizou-se o envoltório geométrico AABB. Foi observado um melhor desempenho quando a célula é maior que o maior objeto e menor que o

tamanho do ambiente, a Figura 9 e a Figura 10 mostram os resultados obtidos.

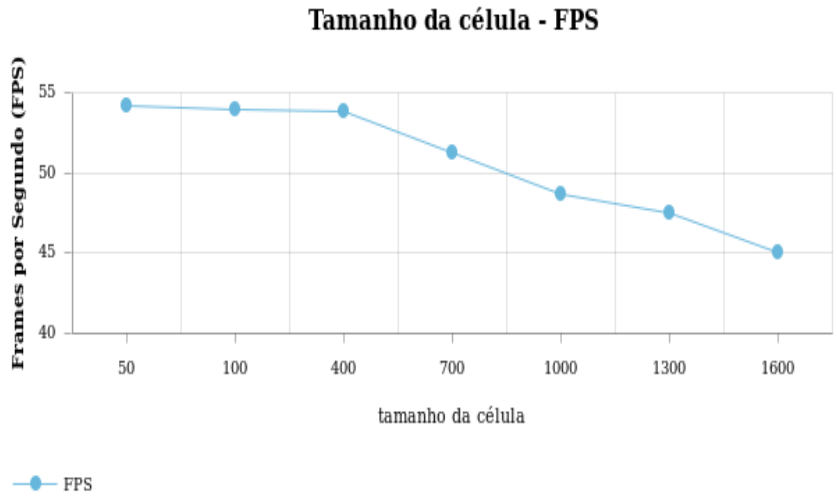


**Figura 9** FPS em relação ao tamanho da célula da Grid

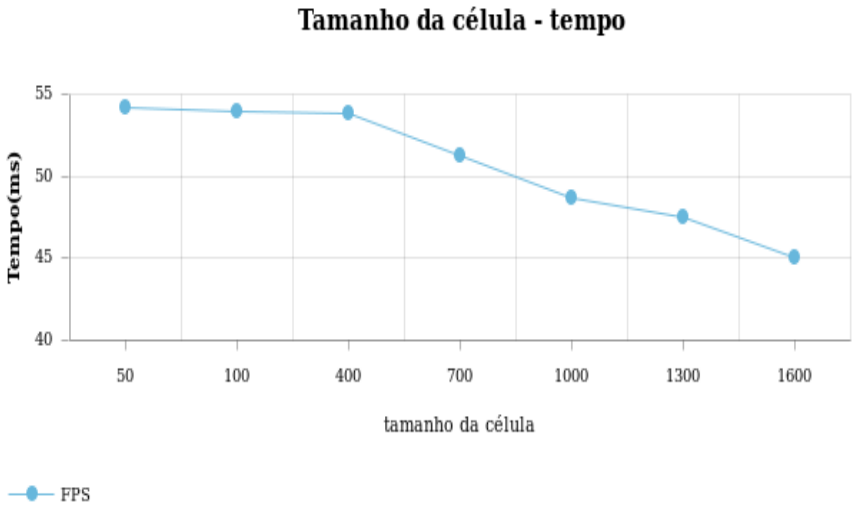


**Figura 10**. Tempo gasto na fase de colisão em relação ao tamanho da célula da Grid

## 4.2. ANÁLISE 2

Na segunda análise foi observada a influência da quantidade de objetos em relação a taxa de frames por segundo (FPS), tempo gasto na fase de detecção de colisão e total de testes de colisão realizados.

Nesta análise também foi feito a comparação entre a técnica de Grid utilizando *Spatial Hashing* e a técnica de Força Bruta.

Os objetos gerados possuíam direções e velocidade aleatórias sendo que a velocidade (pixels por segundo) e o tamanho (pixels) dos objetos variavam de 5 a 50.

Nas Figuras 11 e 12 é apresentado a influência da quantidade de objetos em relação a taxa de frames por segundo (FPS), comparando a técnica Força Bruta com a técnica de Grid *Spatial Hashing*, sendo que na Figura 11 foram criados objetos com

envoltório geométrico AABB e na Figura 12 envoltório geométrico esfera.

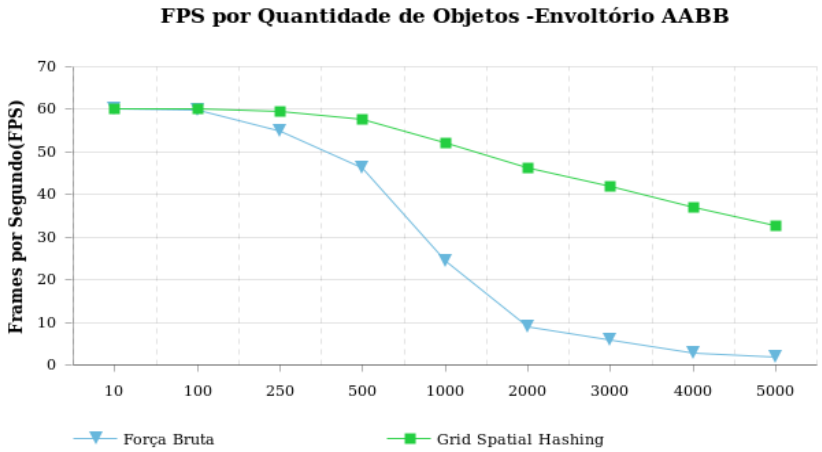


**Figura 11**. Taxa de frames por segundo (FPS) por quantidade de objetos com envoltório AABB, utilizando Força Bruta e Grid.

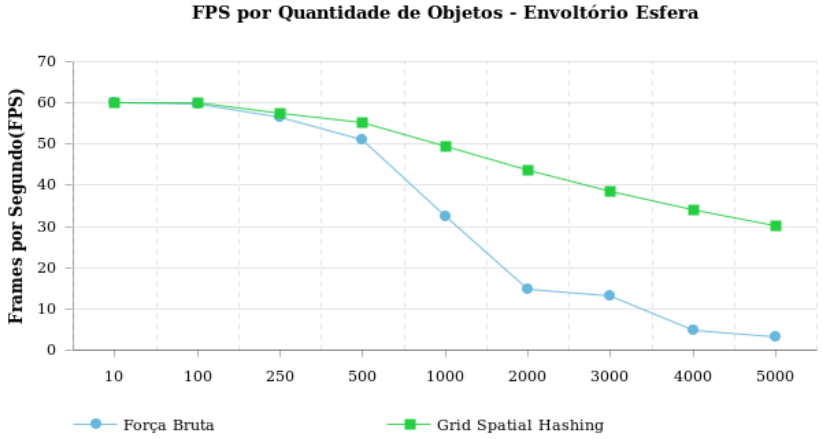


**Figura 12.** Taxa de frames por segundo (FPS) por quantidade de objetos com envoltório esfera, utilizando Força Bruta e Grid.

Nas Tabelas 1 e 2 é apresentado a comparação da técnica Grid com a técnica Força Bruta, nos resultados do tempo gasto na fase de detecção de colisão e o total de testes de colisão realizados a cada frame, sendo que estes valores são a média da execução de 500 frames. Na Tabela 1 foram utilizados objetos com envoltório AABB e na Tabela 2 objetos com envoltório esfera.

**Tabela 1.** Resultados de total de testes por frame e tempo gasto na fase de colisão com técnica Grid e técnica Força Bruta, utilizando objetos com envoltório AABB.

| **Envoltório AABB** | **Técnica de Grid** | | **Técnica Força Bruta** | |
|---|---|---|---|---|
| **Total de Objetos** | **Total de testes de colisão por frame** | **Tempo (ms) fase colisão** | **Total de teste de colisão por frame** | **Tempo (ms) fase colisão** |

| | | | | |
|---|---|---|---|---|
| 10 | 0,95 | 0,01 | 45 | 0,01 |
| 100 | 37,72 | 0,11 | 4950 | 0,27 |
| 250 | 193,76 | 0,14 | 31125 | 1,69 |
| 500 | 787,09 | 0,34 | 124750 | 4,78 |
| 1000 | 3116,15 | 1,77 | 499500 | 23,51 |
| 2000 | 13035,04 | 3,20 | 1999000 | 91,92 |
| 3000 | 29347,91 | 4,27 | 4498500 | 151,19 |
| 4000 | 52918,48 | 6,62 | 7998000 | 354,80 |
| 5000 | 82749,01 | 9,13 | 12497500 | 567,86 |

**Tabela 2.** Resultados de total de testes por frame e tempo gasto na fase de colisão com técnica Grid e técnica Força Bruta, utilizando objetos com envoltório esfera.

| **Envoltório Esfera** | **Técnica de Grid** | | **Técnica Força Bruta** | |
|---|---|---|---|---|
| **Total de Objetos** | **Total de teste de colisão por frame** | **Tempo (ms) fase colisão** | **Total de teste de colisão por frame** | **Tempo (ms) fase colisão** |
| 10 | 1,54 | 0,02 | 45 | 0,05 |
| 100 | 33,86 | 0,554 | 4950 | 0,104 |
| 250 | 179,5 | 0,119 | 31125 | 0,62 |
| 500 | 735,9 | 0,211 | 124750 | 2,14 |
| 1000 | 2900,18 | 1,36 | 499500 | 12,22 |
| 2000 | 11661,21 | 1,94 | 1999000 | 46,5 |
| 3000 | 26367,18 | 2,6 | 4498500 | 53,82 |
| 4000 | 47369,48 | 3,82 | 7998000 | 179,2 |
| 5000 | 73883,62 | 4,85 | 12497500 | 284,97 |

A técnica de Grid com *Spatial Hashing* obteve melhores resultados do que a técnica de Força Bruta, observou-se que quando a quantidade de objetos é menor, as técnicas não possuem muita diferença entre resultados obtidos, mas quando quantidade de objetos aumenta consideravelmente a técnica de Grid tem um melhor resultado em relação a técnica de Força Bruta.

O tipo de envoltório esfera e envoltório AABB tiveram resultados muito próximos, sendo que o envoltório esfera obteve uma pequena vantagem no tempo gasto na fase de colisão, em ambas as técnicas analisadas.

## 5. CONSIDERAÇÕES FINAIS

Neste trabalho foram analisadas as técnicas de colisão Grid com *Spatial Hashing* e a técnica Força Bruta em um ambiente Web, os resultados obtidos nos experimentos realizados demonstraram que a técnica de Grid possui um melhor desempenho de que a de Força Bruta e que o tipo de envoltório AABB obteve uma pequena vantagem no tempo gasto na fase de colisão entre poucos objetos, sendo que o envoltório tipo Esfera observou-se um desempenho maior com muitos objetos. Na técnica de Grid foi observado um melhor desempenho quando o tamanho da célula é maior que o maior objeto e menor que o tamanho do ambiente.

Mesmo com uma grande quantidade de objetos, aplicando a técnica de colisão por Grid com *Spatial Hashing*, foi obtido uma experiência satisfatória num ambiente Web.

## 6. REFERÊNCIAS


ERICSON, C**. Real-Time collision detection**. Morgan Kaufmann Publishers, 2005.

HASTINGS, E. J.; MESIT, J.; GUHA, R. K. **Optimization of large-scale, real-time simulations by spatial hashing**. 2005. Disponível em: <http://www.eecs.ucf.edu/~jmesit/publications/scsc%202005.pdf>. Acesso em: 16 abr. 2015.

HASTINGS, E. J.; MESIT, J.; GUHA, R. K. **T-Collide: a temporal, real-time collision detection technique for bounded objects**. 2004. Disponível em: <http://www.cs.ucf.edu/~jmesit/publications/T-Collide%20CGAIDE%202004.pdf>. Acesso em: 15 jul. 2015.

KERA, M.; PEDRINI, H.; NUNES, F. **Detecção de colisões em ambientes virtuais para treinamento médico.** In: Workshop de Realidade Virtual e Aumentada, nov. 2007. Disponível em: <http://www.seer.ufrgs.br/rita/article/download/rita_v18_n2_p205/12810>. Acesso em: 16 abr. 2015.

ROCHA, R. S.; RODRIGUES, M. A. F. Análise de Desempenho de Algoritmos para Detecção de Colisão em Ambientes Gráficos Interativos. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 17. **Anais...** Rio de Janeiro, 2007.

SANTOS, R. G. **Detecção de colisão para um simulador de robô manipulador.** 2007. 124p. Dissertação (Mestrado) - Universidade do Estado de Santa Catarina, 2007.

SCHORNBAUM, F. **Hierarchical hash grids for coarse collision detection.** 2009. Disponível em: <https://www10.informatik.uni-erlangen.de/~schornbaum/hierarchical_hash_grids.pdf>. Acesso em: 22 abr. 2015.

SOUZA, M. S. **Simulação interativa de tecidos e roupas: técnicas para o desenvolvimento de um simulador.** 2014. 77p. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, 2014.

TADDEO, L. S. **Detecção de colisão utilizando Grids e Octrees esféricas para ambientes gráficos interativos**. 2005. 103p. Dissertação (Mestrado) - Universidade de Fortaleza (UNIFOR), Ceará, 2005.